GitHub

Un lugar en la nube donde podes alojar tus proyectos de programacion. Estables,

funcionales y rapidos.

Repositorios: Lugar donde se iran almacenando los archivos de tu proyecto y podras

hacer el seguimiento de los mismos. Un repositorio por proyecto.

-Repositorios remotos: Estan en la nube. Para IT repo remoto se denomina Origin.

-Repositorios locales: Cada persona de tu equipo tiene una copia de ese repositorio

en su computadora.

Hay que generar un vinculo entre los repositorios locales y remoto.

En GitHub vamos alojar una version online y compartida de nuestros archivos, cada persona de

nuestro equipo a traves de los comandos de git se bajara los cambios y subira

sus actualizaciones.

-Conectar el repositorio local con el repositorio remoto. Cuando creas un repositorio en

GitHub se genera una URL unica. En la terminal escribir:

\*git remote add origin URL (copiado de GitHub, indicarle con que repositorio remoto sincronizar)

\*git remote -v (te indica los repositorios locales y remoto sincronizados)

-Para subir archivos al repositorio remoto desde el repositorio local(tener todo commiteado):

\*git push origin master (o main) (envia los archivos que tenes en tu repositorio local al remoto,

en la rama principal que se llama master o main.)

Ramas: una rama dentro de un repositorio es una copia alternativa del mismo. Linea paralela.

Agregar nuevas funcionalidades sin tener que modificar la linea original del tiempo.

Una version2 de tu proyecto. Y despues fusionarlo con la rama principal (master o main)

-Para bajar archivos del repositorio remoto(Github):

\*git clone URL exacta (ejecutar una sola vez sobre la carpeta en tu dispositivo en la que queres

descargar, y que los archivos que vas a descargar no esten en nuestro dispositivo)

\*git pull origin master (o main)(bajar y actualiz los archivos que sufrieron alguna modificacion)

-Resolviendo conflictos:

Cuando hay un conflicto cuando quiero subir mi archivo en la nube con:

\*git push origin master

es porque mas de uno modifico el mismo archivo, entonces es cuando intento bajar lo de la nube con:

\*git pull origin master

si salta error, entonces lo que salga arriba van a ser mis cambios y abajo mi companero.

<<<<<<<HEAD aca va a decir lo que yo hice

Mis otros cambios

=======

Mis cambios

>>>>>>> 3216f3d5ca65cfd3252ae76808d8f659a7 y aca lo que hizo mi companero

Dejar la parte del codigo que decidimos, y :

\*git add nombreArchivo.js

\*git commit -m "Resolvi el conflicto en nombreArchivo.js"

\*add push origin master

para reducir conflictos:

-tener commits pequenos y subir al servidor frecuentemente.

-uso de ramas para trabajar en paralelo a la version principal del codigo.

\*git rm -r directorio (remueve el directorio y los archivos que contiene)

\*git log -- <ruta del archivo> (muestra el historial de un archivo específico)

\*git log --author=usuario (muestra el historial de un usuario en particular)

\*git checkout -- archivo (solo debe usarse mientras el archivo no se haya

añadido todavía al área de trabajo temporal)

\*git reset HEAD archivo (debe usarse cuando el archivo ya ha sido añadido

en el área temporal)

\*git checkout nombre\_archivo (permite realizar el cambio de directorio)

\*git remote

\*git remote -v

\*git remote add origin git@github.com:minombre/archivo-git.git (enlaza el

repositorio local con un repositorio remoto)

\*git remote show origin (permite ver la información de los repositorios

remotos)

\*git remote rename origin nombre\_nuevo (renombra un repositorio

remoto)

\*git remote rm nombre\_git (desvincula un repositorio remoto)

\*git push -u origin master (el primer push en el repositorio debe contener

su nombre y branch)

\*git push (los otros pushs no necesitan otras informaciones)

Branch: El master es la branch principal de Git.

El HEAD es un puntero especial que indica cual es la branch actual. Por defecto,

HEAD apunta a la branch principal, la master.

\*git branch nuevaBranch\_nombre (crea una nueva branch)

\*git checkout nuevaBranch\_nombre (cambia a una branch existente) - En este

caso, el principal puntero HEAD está apuntando a la branch llamada

nuevaBranch\_nombre.

\*git checkout -b nuevaBranch\_nombre (crea una nueva branch y apunta a ella)

\*git checkout master (vuelve a la branch principal-master-)

\*git merge nuevaBranch\_nombre (resuelve la unión (merge) entre las branches) -

Para realizar la unión (merge), debe estar en la branch que debe recibir los

cambios.(es decir, en la master)

\*git branch -d nuevaBranch\_nombre (apagando una branch)

\*git branch (lista branches)

\*git branch -v (lista branches con información de los últimos commits)

\*git branch --merged (lista branches que ya se han unido (merged) con la master)

\*git branch --no-merged (listar branches que no se han unido (merged) con la

master)

\*git pull origin nombreeBranch (saca los archivos de una branch existente)

\*git push origin nuevaBranch\_nombre (crea una branch remota con el mismo

nombre)

\*git merge --abort o git reset --merge (cuando tenemos problemas con la unión

(merge) y queremos deshacerla)

\*git reset HEAD (cuando queremos volver a un commit anterior, si queremos

volver a más de un commit, debemos poner el número de commits después de

HE AD. Ejemplo: HEAD~2)